Charlie Basantes

COP 4045 001

4/5/2020

Homework #5

1. Starting from a “seed” value X0 the pseudo-random number sequence is defined by this formula: Xn+1=(aXn+c)modm , where m is a positive integer called the modulus, a is multiplier (0 < a < m), and c is the increment (0 ≤ c < m). In addition, 0 ≤ X < m. In Python, the mod operator is %.

For this problem we use the following values:
m = 232
a = 22695477
c = 1.

Part a). Iterator Class Write a class called RndSeq that generates the sequence of n pseudo-random numbers starting with a seed value x0, where n and x0 are given as parameter to the constructor. If n < 0 the object generates an infinite sequence of numbers. The class must comply with the iterator interface (__iter__) and must support the for statement. The __next__() method must return the next pseudo-random number from the sequence or throw StopIteration if n numbers have already been generated.

This class should work like this:

```
 >>> rnd = RndSeq(1, 10)
>>> [i for i in rnd]
 [22695478, 2156045615, 2867233980, 71484141, 2911408402, 2613937339, 1153135800, 420428313, 1503962414, 4187371143]
```

… or like this:

```
[22695478, 2156045615, 2867233980, 71484141, 2911408402, 2613937339, 1153135800, 420428313,
 >>> rnd = RndSeq(1, 2)     # generate two pseudo-random numbers
 >>> it = iter(rnd)
>>> next(it) 22695478
>>> next(it) 2156045615
>>> next(it)
 Traceback (most recent call last):
          File "<pyshell#190>", line 1, in <module>
                    next(it)
          File "……/1.py", line 207, in __next__
                    raise StopIteration
         StopIteration
```

Part b). Generator Write a Python generator called rnd_gen(x0, n) that takes a positive integer seed x0 and an int n and produces the sequence of the first n pseudo-random numbers (if n ≥ 0) or an infinite sequence of numbers (if n < 0). This generator is defined as a function that uses the yield keyword to output a value, as seen on the Chapter 16 lecture PDF file, on slides 60-70. This generator produces the same number sequence as the RndSeq class from part a). Here is how it can be used in a for loop to print the first 10 pseudo-random numbers with seed 1:

>>> [i for i in rnd_gen(1, 10)]

[22695478, 2156045615, 2867233980, 71484141, 2911408402, 2613937339, 1153135800, 420428313, 1503962414, 4187371143]

>>> list(rnd_gen(1, 3))

[22695478, 2156045615, 2867233980]

>>>

Add in this file a function called main() that demonstrates both the class and the generator by creating and printing lists with the first 10 prime numbers with seed 2.

Take a screenshot with the output of the main() function and insert in the PDF file.

My Solution:

```
# -*- coding: utf-8 -*-
"""
Created on Sun Apr  1 15:47:13 2020

@author: solid
"""
#import math

#Part A
class RndSeq:

    def __init__(self, x0, n):
        self.x0 = x0
        self.n = n
        self.m = pow(2,32)
        self.a = 22695477
        self.c = 1
        self.count = 0

    def  __iter__(self):
        self.x = self.x0
        return self

    def __next__(self):
```

```
        #Should run infinitly if n is a negative number
        self.count += 1
        if self.count > self.n and self.n >= 0:
            raise StopIteration
        else:
            self.x = (self.a * self.x + self.c) % self.m

        return self.x

#Part B
def rnd_gen(x0, n):
    m = pow(2,32)
    a = 22695477
    c = 1
    x = x0
    count = 0

    #runs infinitly if n is negative
    while count < n or count > n:
        x = (a * x + c) % m
        yield x
        count += 1

print("Class RndSeq Sequence method: ")
rnd = RndSeq(1, 10)
print([i for i in rnd])
print()
print('-' * 20)

print("Generator rnd_gen method: ")
print([i for i in rnd_gen(1, 10)])
print(list(rnd_gen(1,3)))
print()
print('-' * 20)

rnd = RndSeq(1,2)
it = iter(rnd)
print(next(it))
print(next(it))
print("if I call next() one more time, program will raise StopIteration and
exit, so I commented it out.")
#print(next(it))


def main():
    print('-' * 20)
    print()
    print("Printing lists with the first 10 random numbers with seed 2")
    print()
    print("Class RndSeq Sequence method: ")
    rnd = RndSeq(2, 10)
```

```
    for i in rnd:
        print(list(rnd))

    print('-' * 20)
    print("Generator rnd_gen method: ")
    print(list(rnd_gen(2,10)))


if __name__ == "__main__":
    # execute only if run as a script
    main()
```

Screenshot:

```
In [48]: runfile('D:/COP4045 Python/Hw5/p1_Basantes_Charlie.py', wdir='D:/COP4045 Python/Hw5')
Class RndSeq Sequence method:
[22695478, 2156045615, 2867233980, 71484141, 2911408402, 2613937339, 1153135800, 420428313, 1503962414, 4187371143]

--------------------
Generator rnd_gen method:
[22695478, 2156045615, 2867233980, 71484141, 2911408402, 2613937339, 1153135800, 420428313, 1503962414, 4187371143]
[22695478, 2156045615, 2867233980]

--------------------
22695478
2156045615
if I call next() one more time, program will raise StopIteration and exit, so I commented it out.
--------------------

Printing lists with the first 10 random numbers with seed 2

Class RndSeq Sequence method:
[45390955, 4289395752, 3578422345, 1570701598, 1456365367, 2316466276, 3987301557, 3982688122, 2587496515]
--------------------
Generator rnd_gen method:
[45390955, 4289395752, 3578422345, 1570701598, 1456365367, 2316466276, 3987301557, 3982688122, 2587496515, 2575812576]
```

2.

a) Write a generator gen_rndtup(m) that creates an infinite sequence of tuples (a, b) where a and b are two random integers obtained using the rnd_gen(1, -1) generator from Problem 2 and $0 \le a \le b < m$. (Hint: Remember that for any positive integers m and q, $0 \le q \% m < m$. If q is random, then q % m is also random.)

b) Write code that uses lambda expressions, the itertools.islice function (https://docs.python.org/3/library/itertools.html#itertools.islice), and the filterfalse function to display the first 8 generated tuples (a, b) from gen_rndtup(10) (from part a) that have a + b >= 6. Example: with n==10 the output could be: (2,5), (5,5), (1,6),(7,9),...

c) Write a for loop using generator expressions and the zip function to display the first 8 tuples (a, b), where a is obtained using generator rnd_gen(1, -1), b is obtained using generator rnd_gen(2, -1), and $0 \le a \le b \le 100$. The idea is to filter out tuples for which a > b.

d) Write code with the gen_rnd(1, -1) generator, lambda expressions, map(), itertools.islice, and the filter functions to display the first 10 random numbers between 0 and 100 that are divisible to 13.

e) Write code with generator gen_rndtup(m=10) from part a), lambda expressions, map(), the itertools.islice, functools.reduce(), and the filter functions to display the sum of first 10 generated tuples (a, b) that have sum a + b >= 5.

The sum of tuples is done component-wise for each tuple element. E.g. if the sequence filtered is (1,5), (2,6), (6,6),(3,5), then the sum of these tuples that is displayed is (1+2+6+3, 1+6+6+5) = (12, 22).

Run the code from parts a) – e) and take one or more screenshots with the output. Include the screenshots in the PDF file.

What I have:

```
# -*- coding: utf-8 -*-
"""
Created on Sun Apr  5 17:31:23 2020

@author: solid
"""
from itertools import islice
from itertools import filterfalse

def rnd_gen(x0, n):
    m = pow(2,32)
    a = 22695477
    c = 1
    x = x0
    count = 0

    while (count < n or count > n) and count != 10:
        x = (a * x + c) % m
        count += 1
        yield x


def gen_rndtup(m):
    q = []
    rnd = rnd_gen(1,-1)
    for i in rnd:
        q.append(i)
    print(list(q))

    for i in range(m):
        q[i] = q[i] % m
```

```
    print(list(q))

    x = filterfalse(lambda a,b : a + b >= 6, range(10))
    print(x)


def main():
   gen_rndtup(10)


if __name__ == "__main__":
    main()
```